Computational Visual Media
DOI 10.1007/s41095-016-0062-4 Vol. 2, No. 4, December 2016, 321–327

Research Article

Computational design of iris folding patterns

Yuki Igarashi[1](✉), **Takeo Igarashi**[2], **and Jun Mitani**[3]

Abstract Iris folding is an art-form consisting of layered strips of paper, forming a spiral pattern behind an aperture, which can be used to make cards and gift tags. This paper describes an interactive computational tool to assist in the design and construction of original iris folding patterns. The design of iris folding patterns is formulated as the calculation of a circumscribed polygonal sequence around a seed polygon. While it is possible to compute the positions of vertices analytically for a regular polygon, it is not straightforward to do so for irregular polygons. We give a numerical method for irregular polygons, which can be applied to arbitrary convex seed polygons. The user can quickly experiment with various patterns using the system prior to constructing the art-form.

Keywords craft; pattern; fabrication; user interface; novice users

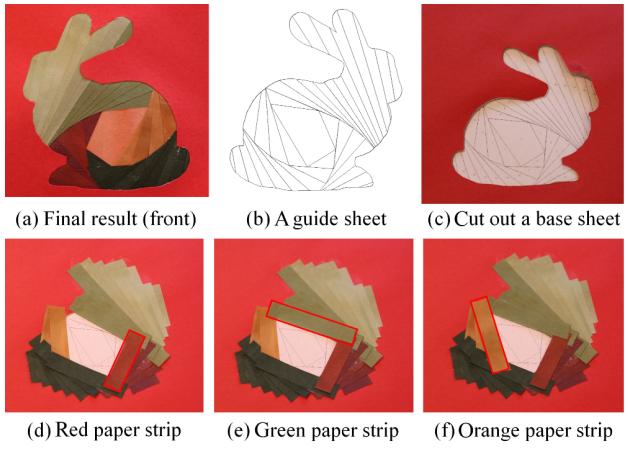


Fig. 1 (a) The end result viewed from the front. (b) A guide sheet. (c)–(f) The construction process viewed from the back. (c) The user first cuts out some paper and turns it over onto a guide sheet. (d)–(f) The user then pastes colored paper strips around the central polygon.

1 Introduction

Iris folding is an art-form consisting of layered paper strips that form a spiral pattern behind an aperture, as shown in Fig. 1. Iris folding is a simple and fun paper-folding technique that can be used to make greeting cards and gift tags. The design forms an iris-like pinhole at the center, similar to that of an eye or a camera lens. In a typical workflow, a person first prepares a guide sheet (usually taken from a book), base sheet, and paper strips. She then cuts the base sheet making a hole as shown in the guide sheet. She places the base sheet on the guide sheet and pastes the paper strips on the base sheet following the instructions shown in the guide sheet. After pasting all the paper strips, she turns the base sheet over to obtain the final result.

There are several interesting geometric properties of iris folding patterns, which make the design of such patterns an intriguing mathematical problem. Although it can be easy to design basic iris folding patterns, even for novices, many of the more complex geometrical patterns must satisfy certain geometric constraints, as shown in Fig. 2. It is typically difficult for novices to design such patterns. If the geometric constraints are not satisfied, paper is wasted because the design does not effectively use the width of the paper strips. In addition, manual construction takes more time as the number of required polygons increases.

This paper presents an interactive computational system to assist in the design and construction of original iris folding patterns that satisfy geometric

1 Meiji University, Nakano-ku, 164-8525, Japan. E-mail: yukim@acm.org (✉).
2 The University of Tokyo, Bunkyo-ku, 113-0033, Japan. E-mail: takeo@acm.org.
3 University of Tsukuba, Tsukuba-city, 305-8573, Japan. E-mail: mitani@cs.tsukuba.ac.jp.
Manuscript received: 2016-08-27; accepted: 2016-09-23

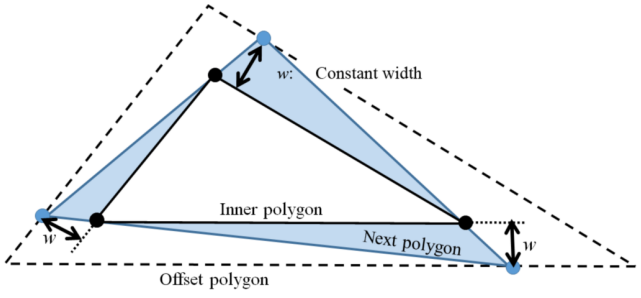


Fig. 2 Problem: calculate the next polygon circumscribing the inner polygon such that the distance from the vertices of the polygon to the corresponding internal polygon edge is constant.

constraints. Figure 3 shows an overview of the process. The user first inputs an outline and a seed polygon inside it. The system then automatically computes an iris folding pattern, i.e., a series of strips around the seed polygon. The user can experiment with arbitrary polygons and color combinations prior to completing the pattern.

The design of iris folding patterns requires the computation of a series of strips of equal width around the polygon. As shown in Fig. 2, we formulate this design problem as the calculation of an offset polygonal sequence around an inner polygon, satisfying the constraint that the distance from the vertices of the polygon to the corresponding internal edge of the polygon must be constant (i.e., the width of the paper strips is constant). In this manner, the system computes iris folding patterns around an arbitrary convex seed polygon.

2 Related work

Various interesting systems have been proposed recently to support the fabrication of physical objects using state-of-the-art graphics techniques. Mitani and Suzuki [1] presented a system for paper craft. Li et al. [2, 3] presented systems for popup cards.

As for two-dimensional objects, Coahranm and Fiume [4] reported a sketch-based design system for a specific quilting art-form, Bargello patterns. They described an algorithm that transforms sketched input data into graceful Bargello curves. Holly [5] is an interactive stencil design system for novices, providing a method for generating expressive stencils, where a user simply uses standard drawing operations and the system automatically generates the appropriate stencil satisfying constraints. Patchy [6] is an interactive system that can assist in the design of original patchwork patterns. The user designs original patchwork strokes, and can quickly experiment with various patterns prior to sewing.

The pursuit curve of a polygon forms a geometric pattern similar to iris folding. It is defined as the trajectories of vertices starting from the corners of the polygon, and moving inwards pursuing the neighboring vertices [7]. In the case of a triangle, three pursuit curves converge to a point known as the Brocard point. Although pursuit curves form similar patterns to those seen in iris folding, the definitions differ, and the analysis of pursuit curves cannot be applied directly to our problem.

3 User interface

Figure 3 shows an overview of the process. The user first inputs an outline and a seed polygon, as shown in Fig. 3(a). Then it automatically computes an iris folding pattern (Fig. 3(b)). The user can experiment with arbitrary polygons and color combinations prior to completing the pattern. The user can also apply textured rendering as shown in Fig. 3(c). The user finally creates the physical pattern (Fig. 3(d)).

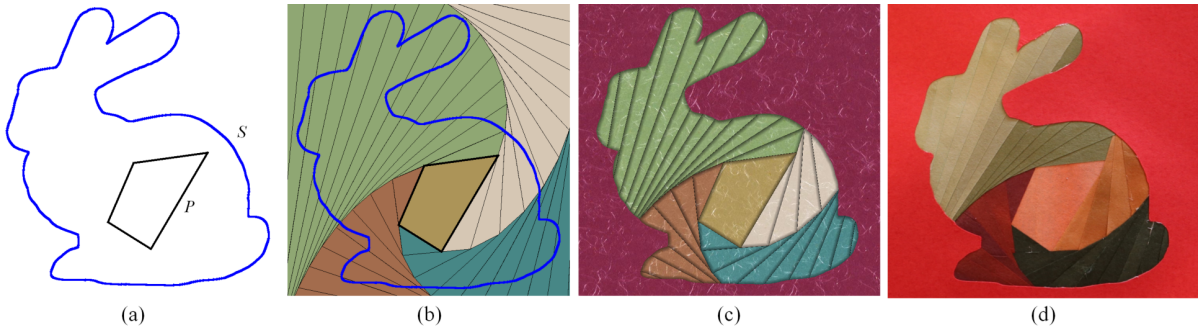


Fig. 3 Overview of our system. (a) The user first inputs an outline S (blue line) and a seed polygon P (black line). (b) Then the system automatically computes a pattern for iris folding and (c) applies texture rendering. (d) The user finally creates the physical pattern.

3.1 User inputs

The user first inputs an outline polygon S and an internal convex seed polygon P inside S. The user can change the width w of the paper strips. The system provides two methods to input the outline S: the user can either draw a stroked line directly, or can load a black-and-white bitmap image and have the system trace the contour line using the marching-squares algorithm [8].

The user draws a polygon P, as shown in Fig. 3(a). Arbitrary polygons can be designed by clicking on desried vertex positions for the polygon. The user can also draw a regular polygon by choosing one from a menu. The system automatically places the polygon at the center, and the user can drag it to a desired position. It is also possible to scale the polygon P. The default width of the paper strips is 10 mm; however, this can be changed using the menu. The user can change the orientation of the strips using the menu.

3.2 Interactive design

Once the outline S, the seed polygon P, and the strip width w have been determined, the system automatically generates a pattern for iris folding. If the user drags the vertices of the internal polygon P, the system updates the pattern in real time. The user can set the colors of the strips using a paint tool. By default, the system shows a design with n-sided polygons with n colors: see Fig. 3(b).

To create a shaded image, the system may also switch to visualization with shading, considering the layers of paper strips, as shown in Fig. 4. Regions of the same color are grouped for previewing, and the system calculates a rendered image for each group. The user can preview the resulting image using a textured image of paper strips. It cannot be manipulated interactively while previewing the texture image, however, and the system switches to normal coloring mode whenever the user performs a dragging operation (Fig. 5).

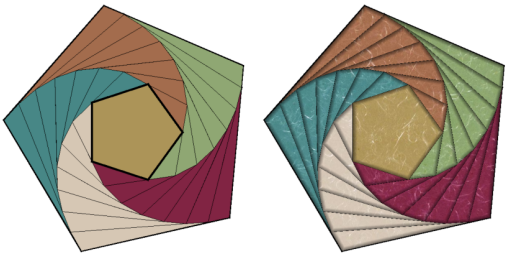


Fig. 4 The fill tool (left) and textured rendering (right).

3.3 Construction guide

The system ultimately outputs a pattern for use in construction. The user pastes paper strips onto the back of the main piece of paper, as shown in Fig. 6. Therefore, the resulting guide sheet is flipped horizontally. The system also shows a step-by-step graphical presentation to aid the construction.

4 Theoretical analysis

This section provides a theoretical analysis of the geometric problem. The algorithm used in the current implementation is described in the next section. First we define the problem, and then we analyze regular seed polygons, followed by irregular seed polygons.

4.1 Definitions

The seed polygon P (input by the user) is defined as P^0. The polygons formed using paper strips are defined as P^m $(m = 1, 2, \dots)$. We consider the problem as determining P^{m+1} from P^m for a given strip width w. The system first computes an offset polygon $P^{m'}$, i.e., the polygon that is obtained by offsetting each edge of P^m by the strip width w.

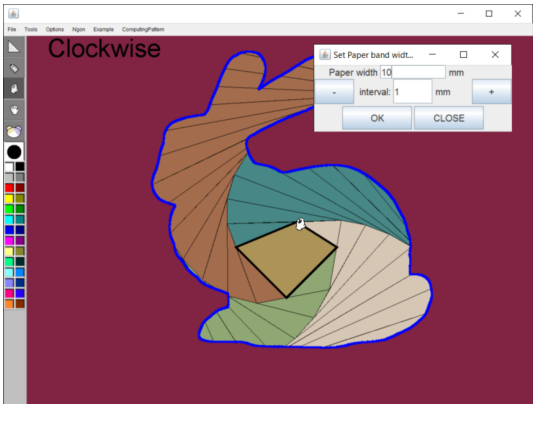


Fig. 5 Screen capture of our system. The user can interactively change the design by dragging the seed polygon.

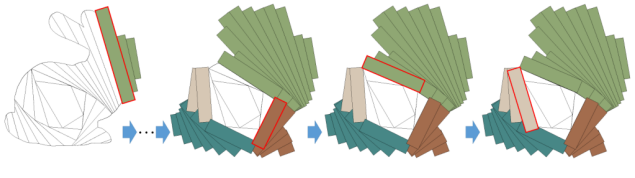


Fig. 6 A step-by-step construction guide.

Then it searches for the P^{m+1} that satisfies the following geometric constraints: (i) the vertices of the next polygon P^{m+1} should be located on the edge of the offset polygon $P^{m'}$, and (ii) the edges of the next polygon P^{m+1} should pass through the vertices of the polygon P^m.

4.2 Analytical solutions for regular polygons

For a regular n-gon (n-sided polygon) P^m, it is possible to analytically calculate the angle of rotation and the scaling ratio that maintain the next polygon P^{m+1} as an n-gon. As shown in Fig. 6, for an internal angle of the n-gon α and strip width w, the edge length l can be found. The system computes the angle of rotation θ whereby the vertices of the polygon P^{m+1} lie on the edges of the strip.

The geometrical relations shown in Fig. 7 can be expressed as follows:

$$\frac{L}{\sin\alpha} = \frac{l}{\sin\theta} = \frac{k}{\sin(\pi-\alpha-\theta)} \tag{1}$$

Eq. (1) can be rewritten as

$$\begin{aligned} w &= k\sin\theta \\ &= L\frac{\sin(\pi-\alpha-\theta)}{\sin\alpha}\sin\theta \\ &= L\frac{\sin(\alpha+\theta)}{\sin\alpha}\sin\theta \end{aligned} \tag{2}$$

When $L = 1$, we obtain the angle of rotation θ as follows:

$$\theta = 2\pi n - 2\arctan\left(\frac{\sqrt{-4w^2+4w\cot\alpha+1}}{2w} - \frac{\sqrt{\frac{\sqrt{-4w^2+4w\cot\alpha+1}}{w^2}+\frac{1}{w^2}+\frac{2\cot\alpha}{w}}}{\sqrt{2}} + \frac{1}{2w}\right) \tag{3}$$

where $\frac{\alpha}{2\pi} - \frac{1}{2} \notin Z$ and $w\tan\left(\frac{\alpha}{2}\right) \neq 0$. The length of a single side of polygon P^{m+1} is given by $L_{m+l} = l + k$.

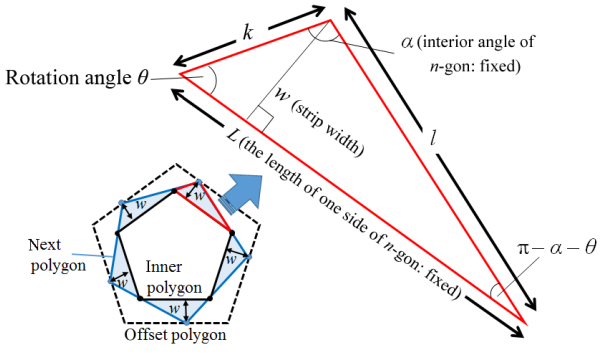


Fig. 7 Iris folding geometry for a regular n-polygon.

4.3 Numerical analysis for arbitrary polygons

Now we describe a numerical approach for an arbitrary n-gon P^m. As shown in Fig. 8, the vertices of an n-gon P^m are defined as $P_0^m, P_1^m, \ldots, P_{n-1}^m$ in clockwise order. We put a point v_{start} on the edge $P_0^{m'}P_1^{m'}$ by varying the parameter t in the range [0, 1]. The straight line l_1 passes through v_{start} and P_1^m, and intersects the edge $P_1^{m'}P_2^{m'}$ at point P_1^{m+1}. The straight line that passes through $P_i^{m'}$ and P_{i+1}^m is line l_{i+1}, which intersects the edge between $P_{i+1}^{m'}P_{i+2}^{m'}$ at P_{i+1}^{m+1}. The straight line that passes through v_{start} and P_0^m is line l_0. The point where lines l_0 and l_{n-1} intersect is $v(t)$. The intersection between the trajectory of $v(t)$ and the edge $P_0^{m'}P_{n-1}^{m'}$ becomes P_{n-1}^{m+1}.

Obtaining an explicit formula to compute P^{m+1} from P^m is overly complicated and difficult. Instead, we analyzed the problem numerically using interactive geometry software (Cinderella [9]). By varying t in the range 0 to 1, the trajectory of $v(t)$ can be found, as shown in Fig. 9. We observe that the trajectories are conic curves, and therefore we can calculate the value t for which the point $v(t)$ is located on an edge of $P^{m'}$ using bisection search. (The Appendix shows that the trajectory is a conic curve, with a quadratic equation, when the seed polygon is a triangle.)

5 Algorithm

We calculate the next polygon P^{m+1} from polygon P^m using bisection search. The system computes the signed distance from $v(t)$ to the corresponding edge $P_0^{m'}P_{n-1}^{m'}$. If the user chooses a clockwise pattern, we search for the parameter t between [0, 0.5]; if the user chooses a counterclockwise pattern, we search

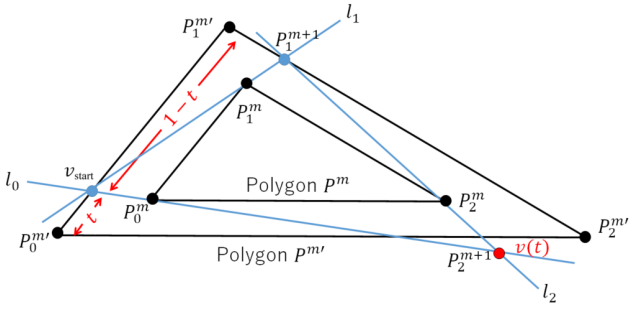


Fig. 8 Analysis of iris folding with arbitrary polygons. The aim is to find t such that $v(t)$ is located on the edge $P_0^{m'}P_{n-1}^{m'}$.

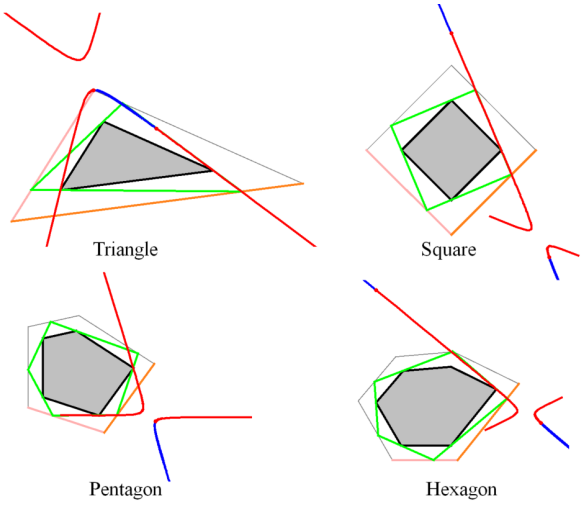


Fig. 9 By varying t from 0 to 1 on the pink edge of the offset polygon, the trajectory $v(t)$ is found (red line). $v(t)$ forms a part of a conic curve: for example, by varying t from -100 to 100, the trajectory $v(t)$ was found (blue line). We find the value t for which the point $v(t)$ is located on the orange edge of the offset polygon.

for t between [0.5, 1]. After finding the value of t, the system generates P^{m+1} using the value. This is repeated until $P^m \supset P(S)$, where $P(S)$ includes the polygon of the desired outline S.

Because the trajectory is a conic curve, a solution does not always exist. The existence of a solution depends on the ratio of the strip width to the edge length of the polygon (see Fig. 10). If a solution cannot be found, the system does not draw a pattern, but instead prompts the user to let the system search for another solution by moving some vertex of the internal polygon, or by changing the strip width w. If a solution still cannot be found, the system tells the user so.

6 Results

A prototype of the system was implemented in Java running on a laptop computer with a 1.2 GHz processor and 2 GB RAM. We used this system to create designs for iris folding patterns, as shown in Fig. 11. Users were allowed to experiment with various patterns using the system prior to beginning construction. A design session typically takes approximately 10–20 minutes, and construction takes about 1 hour.

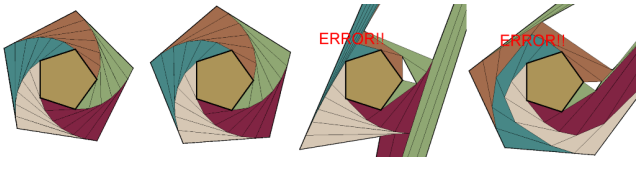


Fig. 10 The existence of a solution depends on the ratio of the strip width to the length of the edges of the polygon. Here the ratio w/l was varied from 0.29 to 0.33, 0.37, and 0.41 (left to right). For $w/l = 0.37$ and 0.41, a pattern could not be formed.

7 Conclusions and future work

We have described an interactive design system for generating iris folding patterns. Users can quickly experiment with various patterns using the system before beginning practical work.

In future, we plan to develop a method to handle multiple internal polygons. We also plan to investigate patterns formed with other constraints, such as symmetry and similarity to the original polygon. We also plan to investigate conditions for the existence of a polygon which circumscribes a polygon and inscribes a polygon. It is also an interesting problem to analyze the requirements concerning the shape of the internal polygon and the upper limit of the ratio of the strip width to the edge length of the polygon for a given shape. We also plan to consider how the system may determine

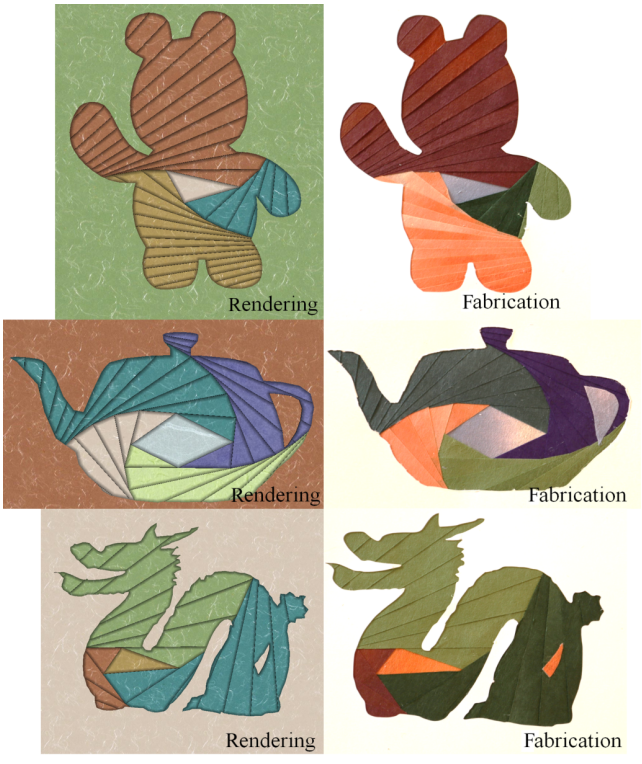


Fig. 11 Design examples created using our system.

which vertices it makes sense for the user to modify, and in which direction it should move, to guide the user to a feasible solution when an error occurs.

Appendix

In this appendix, we show that the trajectory of Q is a conic curve (quadratic equation) when the seed polygon is a triangle.

The vertices of a polygon P^m are defined as P_0^m, P_1^m, P_2^m in clockwise order, as shown in Fig. 12. The system first computes an offset polygon $P^{m'}$, i.e., the polygon that is obtained by offsetting each edge of P^m by the strip width w. We put a point Q_0^{m+1} on the edge $P_0^{m'}P_1^{m'}$. The straight line l_1 passes through Q_0^{m+1} and P_1^m, and intersects the edge $P_1^{m'}P_2^{m'}$ at point Q_1^{m+1}. The straight line l_2 passes through P_1^{m+1} and P_2^m. The straight line that passes through Q_0^{m+1} and P_0^m is line l_0. The point where lines l_0 and l_2 intersect is Q_2^{m+1}. The intersection between trajectory of Q_2^{m+1} and the edge $P_0^{m'}P_2^{m'}$ becomes P_2^{m+1}.

The inner polygon

$$P^m = \{P_0^m(X_0, Y_0), P_1^m(X_1, Y_1), P_2^m(X_2, Y_2)\}$$

is given as input. We observe how $Q_2^{m+1}(x_2, y_2)$ moves as we move $Q_0^{m+1}(x_0, y_0)$ on $P_0^{m'}P_1^{m'}$.

In the following derivation, each occurrence of C represents a different constant value.

We have five constraints as follows:

- $Q_0^{m+1}(x_0, y_0)$ is on the straight line with fixed distance (w) from the edge
$$P_0^m(X_0, Y_0) - P_1^m(X_1, Y_1) \tag{4}$$
- $Q_1^{m+1}(x_1, y_1)$ is on the straight line with fixed distance from the edge
$$P_1^m(X_1, Y_1) - P_2^m(X_2, Y_2) \tag{5}$$
- $P_0^m(X_0, Y_0)$ lies on the edge
$$Q_0^{m+1}(x_0, y_0) - Q_2^{m+1}(x_2, y_2) \tag{6}$$

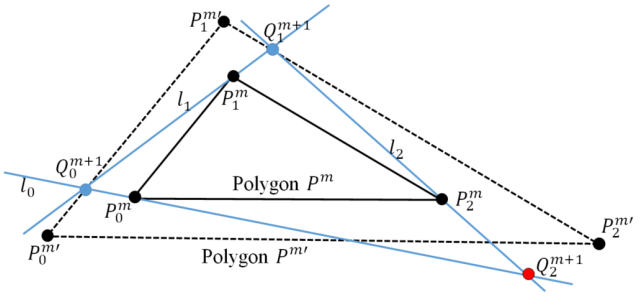


Fig. 12 Inner polygon P^m is given (input) and we observe how $Q_2^{m+1}(x_2, y_2)$ moves as we move $Q_0^{m+1}(x_0, y_0)$ on $P_0^{m'}P_1^{m'}$.

- $P_1^m(X_1, Y_1)$ lies on the edge
$$Q_1^{m+1}(x_1, y_1) - Q_0^{m+1}(x_0, y_0) \tag{7}$$
- $P_2^m(X_2, Y_2)$ lies on the edge
$$Q_1^{m+1}(x_1, y_1) - Q_2^{m+1}(x_2, y_2) \tag{8}$$

We have six unknown values (x_0, y_0), (x_1, y_1), (x_2, y_2). We wish to get an expression relating x_2 and y_2 by eliminating the variables (x_0, y_0) and (x_1, y_1).

Constraint (4) can be rewritten as

$$y_0 = Cx_0 + C \tag{9}$$

[y_0 linearly depends on x_0]

Constraint (5) can be rewritten as

$$y_1 = Cx_1 + C \tag{10}$$

[y_1 linearly depends on x_1]

Constraint (6) can be rewritten as

$$(x_0 - X_0)(y_2 - Y_0) = (y_0 - Y_0)(x_2 - X_0) \tag{11}$$

Constraint (7) can be rewritten as

$$(x_1 - X_1)(y_0 - Y_1) = (y_1 - Y_1)(x_0 - X_2) \tag{12}$$

Constraint (8) can be rewritten as

$$(x_1 - X_2)(y_2 - Y_2) = (y_1 - Y_2)(x_2 - X_2) \tag{13}$$

To eliminate y_0 and y_1, substitute Eqs. (9) and (10) into Eq. (11):

$$(x_0 - X_0)(y_2 - Y_0) = (Cx_0 + C)(x_2 - X_0) \tag{14}$$

To eliminate y_0 and y_1, substitute Eqs. (9) and (10) into Eq. (12):

$$(x_0 - X_1)(Cx_0 + C) = (Cx_1 + C)(x_0 - X_1) \tag{15}$$

To eliminate y_0 and y_1, substitute Eqs. (9) and (10) into Eq. (13):

$$(x_1 - X_2)(y_2 - Y_2) = (Cx_1 + C)(x_2 - X_2) \tag{16}$$

Eq. (14) can be rewritten as

$$x_0 = \frac{Cx_2 + Cy_2 + C}{Cx_2 + Cy_2 + C} \tag{17}$$

$$\left[x_0 \text{ is } \frac{\text{a linear combination of } x_2 \text{ and } y_2}{\text{a linear combination of } x_2 \text{ and } y_2}\right]$$

Eq. (16) can be rewritten as

$$x_1 = \frac{Cx_2 + Cy_2 + C}{Cx_2 + Cy_2 + C} \tag{18}$$

$$\left[x_1 \text{ is } \frac{\text{a linear combination of } x_2 \text{ and } y_2}{\text{a linear combination of } x_2 \text{ and } y_2}\right]$$

To eliminate x_0 and x_1 we substitute Eqs. (17) and (18) into Eq. (14). We first rewrite Eq. (15) as

$$Cx_0x_1 + Cx_0 + Cx_1 + C = 0 \tag{19}$$

Eq. (17) can be rewritten as

$$x_0 = \frac{a(x_2, y_2)}{b(x_2, y_2)} \tag{20}$$

Eq. (18) can be rewritten as

$$x_1 = \frac{c(x_2, y_2)}{d(x_2, y_2)} \tag{21}$$

(a, b, c, d are linear combinations of x_2 and y_2: $a(x_2, y_2) = Cx_2 + Cy_2 + C$).

We substitute Eqs. (20) and (21) into Eq. (19), then multiply both sides by $b(x_2, y_2)d(x_2, y_2)$.

$$Ca(x_2, y_2)c(x_2, y_2) + Ca(x_2, y_2)d(x_2, y_2) + Cc(x_2, y_2)b(x_2, y_2) + Cb(x_2, y_2)d(x_2, y_2) = 0$$

This shows that the trajectory is a conic curve (quadratic equation in x_2 and y_2) because all terms mutliply a linear combination of x_2 and y_2 by a linear combination of x_2 and y_2.

Acknowledgements

We thank Kazushi Ahara for his comments. We also thank Takuya Sawada for his help in writing the paper. This work was supported in part by JSPS KAKENHI Grant Number 26240027.

Electronic Supplementary Material Supplementary material is available in the online version of this article at http://dx.doi.org/10.1007/s41095-016-0062-4.

References

[1] Mitani, J.; Suzuki, H. Making papercraft toys from meshes using strip-based approximate unfolding. *ACM Transactions on Graphics* Vol. 23, No. 3, 259–263, 2004.

[2] Li, X.-Y.; Shen, C.-H.; Huang, S.-S.; Ju, T.; Hu, S.-M. Popup: Automatic paper architectures from 3D models. *ACM Transactions on Graphics* Vol. 29, No. 4, Article No. 111, 2010.

[3] Li, X.-Y.; Ju, T.; Gu, Y.; Hu, S.-M. A geometric study of v-style pop-ups: Theories and algorithms. *ACM Transactions on Graphics* Vol. 30, No. 4, Article No. 98, 2011.

[4] Coahranm, M.; Fiume, E. Sketch-based design for Bargello quilts. In: Proceedings of Eurographics Workshop on Sketch-Based Interfaces and Modeling, 165–174, 2005.

[5] Igarashi, Y.; Igarashi, T. Holly: A drawing editor for designing stencils. *IEEE Computer Graphics and Applications* Vol. 30, No. 4, 8–14, 2010.

[6] Igarashi, Y.; Mitani, J. Patchy: An interactive patchwork design system. In: Proceedings of ACM SIGGRAPH 2015 Posters, Article No. 10, 2015.

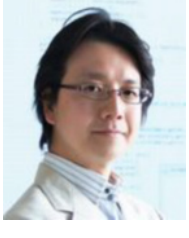
[7] Peterson, I. Pursuing pursuit curves. 2001. Available at https://www.sciencenews.org/article/pursuing-pursuit-curves.

[8] Lorensen, W. E.; Cline, H. E. Marching cubes: A high-resolution 3D surface construction algorithm. *ACM SIGGRAPH Computer Graphics* Vol. 21, No. 4, 163–169, 1987.

[9] Cinderella. The interactive geometry software. 1998. Available at http://cinderella.de/.

Yuki Igarashi is a senior assistant professor in interdisciplinary mathematical science at Meiji University. She received her Ph.D. degree from the Department of Engineering at the University of Tokyo in 2010. From 2010 to 2015, she was a JSPS research fellow at University of Tsukuba. Her research interests are in computer graphics and user interfaces.

Takeo Igarashi is a professor in the CS Department at the University of Tokyo. He received his Ph.D. degree from the Department of Information Engineering at the University of Tokyo in 2000. His research interest is in user interfaces in general and his current focus is on interaction techniques for 3D graphics. He is known as the inventor of the sketch-based modeling system called Teddy, and received the Significant New Researcher Award at SIGGRAPH 2006.



Jun Mitani is a professor at the University of Tsukuba. He received his Ph.D. degree in engineering from the University of Tokyo in 2004. He has been a professor at the University of Tsukuba since April 2015. His research interests are centered on computer graphics, especially geometric modeling techniques. He studies the geometry of curved origami as well as interactive design interfaces.